Vertaalregels voor het generen van StUF-basisschema’s vanuit het UGM

# Introductie

To do:

* Naamgevingsconventies voor de te genereren files
* Checken wat er gebeurt met de gegenereerde schema’s als de nieuwe stuf0302.xsd eronder gezet wordt…
* Metagegevens in stuf0302.xsd bevatten nillables en exact attribute. Moet nog worden gecorrigeerd.
* Uitzoeken wat de namespace van aanvullendeElementen is
* Niet meer reffen van attributes behalve entiteittype. Voor noValue is dit al gedaan
* Includen van metagegeven gebeurtenis conditioneel maken. Hiervoor moet het MIG worden uitgebreid met een tagged value. Of moet dit metagegeven pas op berichtniveau worden toegevoegd?
* [RFC0436](https://discussie.kinggemeenten.nl/discussie/gemma/stuf-bg-310/uitbreiding-waardenbereik-aanduidinginhoudingvermissing) (Dynamische waardenlijsten) is nog niet verwerkt in het algoritme.
* Sorteringen wel of niet onderdeel van basisschema’s?
* Voor enumeraties nog niet duidelijk of de naam of de alias (code) gegenereerd moet worden naar schema.
* Algoritme uitbreiden met proxy-constructie

# Notatie

In dit document worden rechte haken […] als placeholder gebruikt om het XSD-template te binden aan het UGM.

|  |  |
| --- | --- |
| M | Waarde van metagegeven M (evaluatie van de expressie M) |
| [ … ] | Verplichte binding naar het UGM |
| [ … ]? | Optionele binding naar het UGM |
| … | … | Of-constructie |
| …\* | Nul of meer keer het voorgaande op een nieuwe regel |
| ( … ) | Haakjes om voorrang te bepalen |

# Algemeen

Er worden twee schema-bestanden gegenereerd:

1. [namespace code]\_ent\_basis.xsd
2. [namespace code]\_datatypes.xsd

De placeholder [namespace code] staat voor de code van de namespace. Bijvoorbeeld de code van de namesapce "http://www.stufstandaarden.nl/basisschema/bg0320" is bg0320.

De vertalingen worden per default in het eerste schema opgenomen tenzij anders vermeld (dan in het tweede schema). Het tweede schema is een include-schema van het eerste schema waarin alle datatypes zijn opgenomen. Het eerste schema bevat louter complexType’s. Deze dienen alfabetisch gesorteerd te zijn. Het tweede schema bevat zowel complexType’s als simpleType’s. Deze dienen allebei apart alfabetisch gesorteerd te zijn: eerst de complexType’s en daarna de simpleType’s.

Genereer binnen de namespace van het gegenereerde basisschema het volgende attribute:

<attribute name="entiteittype" type="[stuf ns prefix]:Entiteittype"/>

In de volgende secties zal met de ref-constructie van XSD naar dit attribute worden verwezen.

# Entiteittype

Een entiteittype uit het UGM wordt vertaald naar twee complexType’s in het XSD-schema. Het eerste complexType (M-basis) vormt de basis en bevat alle elementen, groepen en relaties van het corresponderende entiteittype. Het tweede complexType (M-matchgegevens) bevat alleen de matchgegevens om entiteiten uniek aan te duiden.

## Basistype

Een entiteittype dat geen super- of subtype is wordt als volgt vertaald naar schema:

<complexType name="[Entiteittype.mnemonic]-basis">

<annotation>

<documentation>[Entiteittype.naam]</documentation>

</annotation>

<sequence>

[(Element | Groep)\*]

[<element name="authentiek" type="[stuf ns prefix]:StatusMetagegeven-basis"

minOccurs="0" maxOccurs="unbounded"/>]?

[<element name="inOnderzoek" type="[stuf ns prefix]:StatusMetagegeven-basis"

minOccurs="0" maxOccurs="unbounded"/>]?

[<element ref="[stuf ns prefix]:tijdvakGeldigheid" minOccurs="0"/>

<element ref="[stuf ns prefix]:tijdstipRegistratie" minOccurs="0"/>

<element ref="[stuf ns prefix]:extraElementen" minOccurs="0"/>

<element ref="[stuf ns prefix]:aanvullendeElementen" minOccurs="0"/>]?

[<element name ="historieMaterieel"

type="[ns prefix]:[Entiteittype.mnemonic]-basis"   
 minOccurs="0"

maxOccurs="unbounded"/>]?

[<element name="historieFormeel"

type="[ns prefix]:[Entiteittype.mnemonic]-basis"   
 minOccurs="0" maxOccurs=”1”/>]?   
 [Relatie\*]

</sequence>

<attribute ref="[ns prefix]:entiteittype" fixed="[Entiteittype.mnemonic]"/>

<attributeGroup ref="[stuf ns prefix]:entiteit"/>

</complexType>

Alleen als één van de elementen of groepen een tagged value ‘Indicatie authentiek’ heeft waarin een waarde is ingevuld (dus niet leeg is), dan wordt het element “authentiek” opgenomen in bovenstaand complexType.

Alleen als één van de elementen of groepen een ‘Indicatie in onderzoek’ gelijk aan ‘Ja’ heeft dan wordt het element “inOnderzoek” opgenomen in bovenstaand complexType. Als de waarde van ‘Indicatie in onderzoek’ gelijk is aan ‘Zie groep’ dan wordt het element “inOnderzoek” alleen opgenomen als de waarde in de groep gelijk aan ‘Ja’ is.

Alleen als het entiteittype geen tabel-entiteit is worden de elementen “StUF:tijdvakGeldigheid”, “StUF:tijdstipRegistratie”, “extraElementen” en “aanvullendeElementen” opgenomen.

Alleen als er op minimaal één van de elementen of groepen materiele historie is gedefinieerd door middel van het attribute meta:materieleHistorie=”true”, dan moet in het bovenstaande schema het “historieMaterieel” aanwezig zijn. Hetzelfde verhaal geldt voor het element “historieFormeel”.

Hieronder een voorbeeld van de vertaling van het entiteittype ‘Besluittype’ (BST).

### Voorbeeld

<complexType name="BST-basis">

<annotation>

<documentation>

Besluittype: Generieke aanduiding van de aard van een besluit.

</documentation>

</annotation>

<sequence>

[(Element | Groep)\*]

<element ref="StUF:tijdvakGeldigheid" minOccurs="0"/>

<element ref="StUF:tijdstipRegistratie" minOccurs="0"/>

<element ref="StUF:extraElementen" minOccurs="0"/>

<element ref="StUF:aanvullendeElementen" minOccurs="0"/>

<element name="historieMaterieel"

type="ztc:BST-basis" minOccurs="0"

maxOccurs="unbounded"/>

<element name="historieFormeel"

type="ztc:BST-basis" minOccurs="0"

maxOccurs="unbounded"/>

[Relatie\*]

</sequence>

<attribute ref="ztc:entiteittype" fixed="BST"/>

<attributeGroup ref="StUF:entiteit"/>

</complexType>

Let op: in dit voorbeeld zijn de placeholders [(Element|Groep)\*] en [Relatie\*] nog niet uitgewerkt omdat ze pas later worden beschreven.

## Matchgegevens

De matchgegevens van een entiteittype worden als volgt vertaald naar StUF:

<complexType name="[Entiteittype.mnemonic]-matchgegevens">

<complexContent>

<restriction base="[ns prefix]:[Entiteittype.mnemonic]-basis">

<sequence>

[(Element | Groep | Relatie)\*]

</sequence>

<attribute ref="[ns prefix]:entiteittype" fixed="[Entiteittype.mnemonic]  
 use="required"/>

<attribute name="scope" type="[stuf ns prefix]:StUFScope"   
 use="prohibited"/>

</restriction>

</complexContent>

</complexType>

De elementen, groepen en relaties in de restriction van het bovenstaande complex type worden gespecificeerd door de tagged value ‘Indicatie kerngegeven’ in EA . Met deze elementen kan een object uniek worden aangeduid. Deze gegevens worden ook wel de matchgegevens genoemd.

Hieronder een voorbeeld hoe de unieke aanduiding van het entiteittype Besluittype wordt vertaald naar StUF.

### Voorbeeld

<complexType name="BST-matchgegevens">

<complexContent>

<restriction base="ztc:BST-basis">

<sequence>

[(Element | Groep | Relatie)\*]

</sequence>

<attribute ref="entiteittype" type="ztc:Entiteittype" fixed="BST"   
 use="required"/>

<attribute name="scope" type="StUF:StUFScope" use="prohibited"/>

</restriction>

</complexContent>

</complexType>

# Tabel-entiteit

In de nieuwe versie van StUF wordt er geen onderscheid meer gemaakt tussen entiteiten en tabel-entiteiten. In het UGM wordt dit onderscheid vooralsnog wel gemaakt. Een <<Tabel-entiteit>> wordt op dezelfde manier vertaald als een <<Entiteittype>>. Dat houdt in dat in het vertaal-template van een entiteittype alle placeholders met de term Entiteittype zijn vervangen door de term Tabel-entiteit.

# Tabel-entiteit-element

Een << element>> in een tabelentiteit wordt op dezelfde manier vertaald als een gewoon element (zie sectie Element).

# Enumeration

Een enumeration uit de package Enumeratiesoort wordt als volgt vertaald naar de file [namespace code]\_datatypes.xsd:

<complexType name="[***Capitalize***(Enumeration.Naam)]-e">

<simpleContent>

<extension base="[ns prefix]:[***Capitalize***(Enumeration.Naam)]">

<attribute name="noValue" type="[stuf ns prefix]:NoValue"/>

</extension>

</simpleContent>

</complexType>

<simpleType name="[***Capitalize***(Enumeration.Naam)]">

<restriction base="string">

<enumeration value="…"/>

<enumeration value="…"/>

…

<enumeration value="…"/>

<enumeration value=""/>[[1]](#footnote-1)

</restriction>

</simpleType>

De functie *Capitalize in bovenstaand* templatezet de eerste letter van de string om in een hoofdletter.

# Complex-datatype

Een complex-datatype wordt als volgt vertaald naar de file [namespace code]\_datatypes.xsd:

***Capitalize***(Complex-datatype.naam)]

***Capitalize***(Complex-datatype.naam)

<complexType name="[***Capitalize***(Complex-datatype.naam)]">

<sequence minOccurs="0">

[Data-element]\*

</sequence>

</complexType>

Een Data-element wordt op dezelfde manier vertaald als een gewoon element, zie sectie Element.

# Union

Een Union wordt als volgt vertaald:

<complexType name="[***Capitalize***(Union.naam)]">

<choice minOccurs="0">

[Union-element]\*

</sequence>

</complexType>

Een Union-element wordt op dezelfde manier vertaald als een gewoon element, zie sectie Element.

# Element

In het UGM kan een element verschillende soorten typen hebben:

* voorgedefinieerd type met eigen leeg-constructie (DATUM, JAAR, etc.)
* voorgedefinieerd type zonder leeg-constructie (POSTCODE)
* referentie naar tabelentiteit
* referentie naar complex-datatype (PuntLijnVlak, PostAdresType, Typebedrag, etc.)
* custom type (AN100, N6, etc.)

In de volgende secties geven we voor elke case een vertaling.

## Case: Type is een voorgedefinieerd type

Als het metagegeven Element.Type gelijk is aan één van de waarden in onderstaande tabel dan wordt er een element gecreëerd omdat er gebruik kan worden gemaakt van een bestaande complex type uit de StUF-onderlaag (zie ‘stuf0302.xsd’).

<element name="[Element.Naam]"   
 type="[***StandaardType***([Element.Type])]"   
 minOccurs="[***IsComplexDataType***()]"   
 maxOccurs="[***upperBound***(Element.indicatieKardinaliteit)]"  
 [meta:materieleHistorie="***Boolean***([Element.indicatieMaterieleHistorie])"]?   
 [meta:formeleHistorie="***Boolean***([Element.indicatieFormeleHistorie])"]?  
/>

De functie *IsComplexDataType()* geeft het getal 1 terug als het element onderdeel is van een complex-datatype en anders geeft het 0 terug.

|  |  |
| --- | --- |
| **Element.Type** | **StandaardType** |
| DATUM | [stuf ns prefix]:Datum-e |
| DATUM? | [stuf ns prefix]:DatumMogelijkOnvolledig-e |
| JAAR | [stuf ns prefix]:Jaar-e |
| JAARMAAND | [stuf ns prefix]:JaarMaand-e |
| DT | [stuf ns prefix]:Tijdstip-e |
| DT? | [stuf ns prefix]:TijdstipMogelijkOnvolledig-e |
| POSTCODE | [stuf ns prefix]:Postcode-e[[2]](#footnote-3) |
| INDIC | [stuf ns prefix]:INDIC-e |

De volgende functies worden gebruikt:

* *UpperBound* geeft de waarde van het veld ‘Upper bound’ onderdeel van het samengestelde veld Multiplicity in EA. In geval de waarde “\*” is in het veld ‘Upperbound’ dan zet de functie dit om in “unbounded”.
* *Boolean*zet volgens de onderstaande tabel de waarde in de eerste kolom om in de boolean waarde in de tweede kolom.

|  |  |
| --- | --- |
| **Waarde** | **Boolean** |
| Ja | true |
| Ja, zie regels | true |
| Nee | false |
| Zie groep | *Boolean* (waarde in groep) |
| N.v.t. | false |

Indien de waarde van de functie *Boolean* evalueert naar false dan worden de attributes meta:materieleHistorie en meta:formeleHistorie niet opgenomen in het schema.

## Case: Type verwijst naar enumeratie

Als het type van het element verwijst naar een enumeratie in de package Enumeratiesoort, dan gaat de vertaling als volgt:

<element name="[Element.Naam]"   
 type="[ns prefix]:***Capitalize***([Element.Type])-e"   
 minOccurs="[***IsComplexDataType***()]"   
 maxOccurs="[***UpperBound***(Element.indicatieKardinaliteit)]"  
 meta:materieleHistorie="[***Boolean***(Element.indicatieMaterieleHistorie)]"   
 meta:formeleHistorie="[***Boolean***(Element.indicatieFormeleHistorie)]"/>

In de type van het element wordt verwezen naar het complexType dat al reeds in sectie Enumeration is gegenereerd.

## Case: Type verwijst naar tabelentiteit

In geval Element.Type verwijst naar een tabelentiteit, genereer het volgende element :

<element name="[Element.Naam]"   
 type="[ns prefix]:[***Capitalize***(***LookUpIdElement***([Element.Type]))]-e"   
 minOccurs="[***IsComplexDataType***()]"   
 maxOccurs="[UpperBound(Element.indicatieKardinaliteit)]"  
 meta:materieleHistorie="[Boolean(Element.indicatieMaterieleHistorie)]"   
 meta:formeleHistorie="[Boolean(Element.indicatieFormeleHistorie)]"

[***IncludeNillable***([***LookUpIdElement***([Element.Type]).Type])]/>

De functie *LookUpIdElement* in bovenstaand template geeft de naam van het uniek identificerende tabel-entiteit-element terug. Op deze manier kunnen we verwijzen naar het type van dit tabel-entiteit-element dat al reeds eerder gegenereerd is (zie sectie Tabel-entiteit-element).

**Let op:** Deze manier van werken gaat goed omdat tot nu toe alle uniek identificerende tabel-entiteit-elementen een custom type (AN3, AN6, N2, etc.) hebben. Het algoritme moet nog worden aangepast om dit voor het algemene geval goed te krijgen.

De functie *IncludeNillable(type)* geeft het attribute ‘nillable=”true”’ terug als *type* gelijk is aan één van de onderstaande formaten:

* N
* N[n]
* N[n],[d]

Anders geeft deze functie niets terug.

## Case: Type verwijst naar complex-datatype

In geval Element.Type verwijst naar een complex-datatype, genereer dan het volgende element:[[3]](#footnote-4)

<element name="[Element.Naam]"   
 type="[ns prefix]:[***Capitalize***([Element.Type])]-e"   
 minOccurs="[***IsComplexDataType***()]"   
 maxOccurs="[***UpperBound***(Element.indicatieKardinaliteit)]"  
 meta:materieleHistorie="[***Boolean***(Element.indicatieMaterieleHistorie)]"   
 meta:formeleHistorie="[***Boolean***(Element.indicatieFormeleHistorie)]"/>

## Case: Type verwijst naar interface

Als het type van het element verwijst naar een component met het stereotype <<interface>> (bijvoorbeeld zoals in de packages Generieke Datatypen en GML3) dan gaat de vertaling als volgt:

<element name="[Element.Naam]"   
 type="[ns prefix]:***Capitalize***([Element.Type])-e"   
 minOccurs="[***IsComplexDataType***()]"   
 maxOccurs="[***UpperBound***(Element.indicatieKardinaliteit)]"  
 meta:materieleHistorie="[***Boolean***(Element.indicatieMaterieleHistorie)]"   
 meta:formeleHistorie="[***Boolean***(Element.indicatieFormeleHistorie)]"/>

<complexType name="[***Capitalize***(Element.Type)]-e">

<simpleContent>

<extension base="[ns prefix]:[***Capitalize***(Element.Type)]">

<attribute name="noValue" type="[stuf ns prefix]:NoValue"/>

</extension>

</simpleContent>

</complexType>

Het complexType waarnaar gerefereerd wordt in de base van de extension is al gegenereerd door het interface. Het interface bepaalt welke namespace er ingevuld moet worden in de prefix [ns prefix]. Voor het UGM BG zijn er momenteel twee soorten interfaces: een intern interface (Generieke Datatypen) en een extern interface (GML3). Voor het interne interface zal de namespace "http://www.stufstandaarden.nl/basisschema/bg0320" worden gebruikt en voor het externe interface "http://www.opengis.net/gml/3.2". Momenteel worden deze namespaces hard-coded in de Imvertor-software gezet. Op termijn zou het mooier zijn als je dit met een tagged value kunt bepalen in het EA-model.

## Else: Custom type

In de andere gevallen gaat de vertaling als volgt:

<element name="[Element.Naam]"   
 type="[ns prefix]:***RemovePrefix***([Element.Naam])-e"   
 minOccurs="[***IsComplexDataType***()]"   
 maxOccurs="[***UpperBound***(Element.indicatieKardinaliteit)]"  
 meta:materieleHistorie="[***Boolean***(Element.indicatieMaterieleHistorie)]"   
 meta:formeleHistorie="[***Boolean***(Element.indicatieFormeleHistorie)]"  
 [***IncludeNillable***([Element.Type])]/>

<complexType name="[***RemovePrefix***(Element.Naam)]-e">

<simpleContent>

<extension base="[ns prefix]:[***RemovePrefix***(Element.Naam)]">

<attribute name="noValue" type="[stuf ns prefix]:NoValue"/>

</extension>

</simpleContent>

</complexType>

<simpleType name="[***RemovePrefix***(Element.Naam)]">

[Element.Type]

</simpleType>

De functie *RemovePrefix(string)* verwijdert indien aanwezig de prefix van de string en zet de eerste letter van de overblijvende string om in een hoofdletter. Voorbeelden:

* *RemovePrefix(“Wpl.woonplaatsidentificatie”) = “Woonplaatsidentificatie”*
* *RemovePrefix(“woonplaatsidentificatie”) = “Woonplaatsidentificatie”*

Het bovenstaande element wordt opgenomen in het schema [namespace code]\_ent\_basis.xsd en het bovenstaande complexType en het simpleType worden opgenomen in de include-file [namespace code]\_datatypes.xsd. Onderstaande tabel specificeert hoe het type van het element in het bovenstaande simpleType vertaald wordt.

|  |  |
| --- | --- |
| **Element.Type** | **XSD-schema** |
| AN | <restriction base="xs:string">  …  </restriction> |
| AN[n] | <restriction base="xs:string">  <maxLength value="[n]"/>  …  </restriction> |
| N | <restriction base="xs:integer">  …  </restriction> |
| N[n] | <restriction base="xs:integer">  <totalDigits value="[n]"/>  …  </restriction> |
| N[n],[d] | <restriction base="xs:decimal">  <totalDigits value="[n+d]"/>  <fractionDigits value="[d]"/>  …  </restriction> |
| TXT | <restriction base="xs:string">  …  </restriction> |
| URI | <restriction base="xs:anyURI">  …  </restriction> |

Onderstaande tabel specificeert hoe de facets bij het formaat van het element in het bovenstaande simpleType vertaald worden.

|  |  |
| --- | --- |
| **Facet** | **XSD-schema** |
| Patroon | <pattern value="( … )?"/> |
| Minimum lengte | <minLength value="…"/> |
| Minimum waarde (inclusief) | <minInclusive value="…"/> |
| Maximum waarde (inclusief) | <maxInclusive value="…"/> |

### Voorbeeld

Hieronder een voorbeeld van de vertaling van het element ‘omschrijvingGeneriek’ met formaat AN80 van het entiteittype ‘Besluittype’.

<element name="omschrijvingGeneriek"   
 type="ztc:OmschrijvingGeneriek-e"   
 minOccurs="0"  
 maxOccurs="1"  
 meta:materieleHistorie="true"/>

<complexType name="OmschrijvingGeneriek-e">

<simpleContent>

<extension base="OmschrijvingGeneriek ">

<attribute name="noValue" type="stuf:NoValue"/>

</extension>

</simpleContent>

</complexType>

<simpleType name="OmschrijvingGeneriek">

<restriction base="string">

<maxLength value="80"/>

</restriction>

</simpleType>

# Groep

Een groep in het UGM wordt als volgt vertaald naar XSD:

<element name="[Groep.naam]"   
 type="[ns prefix]:[***Capitalize***(Groep.naam)Grp-basis]"   
 minOccurs="0"   
 maxOccurs="[***UpperBound***(Groep.indicatieKardinaliteit)]"  
 [meta:materieleHistorie="[***Boolean***(Groep.indicatieMaterieleHistorie)]"]?   
 [meta:formeleHistorie="[***Boolean***(Groep.indicatieFormeleHistorie)]"]?  
/>

<complexType name="[***Capitalize***(Groep.naam)]Grp-basis">

<sequence minOccurs="0">

[( Element | Groep | Relatie )\*]

</sequence>

</complexType>

In StUF beginnen de namen van een complex type altijd met een hoofdletter. Hieronder een voorbeeld van de vertaling van de groep Checklistitem naar StUF.

## Voorbeeld

<element name="checklistitem"   
 type="ztc:ChecklistitemGrp-basis"

minOccurs="0"

maxOccurs="unbounded"/>

<complexType name="ChecklistitemGrp-basis">

<annotation>

<documentation>

Te controleren aandachtspunt voorafgaand aan het bereiken   
 van een status van het STATUSTYPE

</documentation>

</annotation>

<sequence>

<element name="naam" type="ztc:Naam-e" minOccurs="0"   
 meta:materieleHistorie="true"/>

<element name="vraag" type="ztc:Vraag-e" minOccurs="0"   
 meta:materieleHistorie="true"/>

<element name="verplicht" type="bg:Indicatie-e"  
 minOccurs="0" meta:materieleHistorie="true"/>

<element name="toelichting" type="ztc:Toelichting-e"

minOccurs="0" meta:materieleHistorie="true"/>

</sequence>

</complexType>

# Relatie

Een relatie wordt als volgt vertaald.

## Element

<element name="[Relatie.naam][Relatie.target.naam]"

type="[ns qualifier]:[Relatie.mnemonic][***SuffixKB***(Relatie)]"

minOccurs="0"

maxOccurs="[***maxVoorkomens***(Relatie.target.indicatieKardinaliteit)]"

[meta:materieleHistorie="[***Boolean***(Relatie.indicatieMaterieleHistorie)]"]?

[meta:formeleHistorie="[***Boolean***(Relatie.indicatieFormeleHistorie)]"]?/>

De functie *SuffixKB(relatie)* geeft de suffix “-matchgegevens” terug als de relatie onderdeel is van een entiteittype met de suffix “-matchgegevens”. Anders geeft deze functie “-basis” terug.

**Let op**: in de definitie van het bovenstaande element is de waarde van minOccurs altijd gelijk aan “0” ongeacht de kardinaliteit die gespecificeerd is in het informatiemodel. In de basisschema’s van StUF zijn alle elementen optioneel. Op die manier wordt hergebruik bevordert. Bijvoorbeeld in vraagberichten wil je eigenschappen van een object die niet relevant zijn voor het stellen van de vraag kunnen weg laten. Of wil je bijvoorbeeld in een mutatiebericht alleen de gegevens opnemen die gewijzigd zijn.

De functie *maxVoorkomens* geeft het maximale aantal voorkomens terug van de kardinaliteit van de doelrol in de relatie (Target Role Multiplicity). Op een analoge wijze geeft de functie *minVoorkomens* het minimale aantal voorkomens terug.[[4]](#footnote-5) De twee functie zijn gedefinieerd in onderstaande tabel.

|  |  |  |  |
| --- | --- | --- | --- |
| **Kardinaliteit  (Multiplicity)** | **minVoorkomens** | **maxVoorkomens  (zonder historie,  formele historie)** | **maxVoorkomens**  **(materiële historie)** |
| \* | 1 | unbounded | unbounded |
| 0 | 0 | 1 | unbounded |
| 0..\* | 0 | unbounded | unbounded |
| 0..1 | 0 | 1 | unbounded |
| 1 | 1 | 1 | unbounded |
| 1..\* | 1 | unbounded | unbounded |
| n..m | n | m | unbounded |

Als we kijken naar de bovenstaande tabel dan zien we dat in geval er materiële is gedefinieerd op de relatie dan geeft de functie *maxVoorkomens* altijd de waarde “unbounded” terug. Dan kan het hierboven gedefinieerde element voor de relatiesoort worden gebruikt voor het opnemen van (meerdere) materiële historische voorkomens.

In het SIM en het UGM is de naam van een relatie niet altijd uniek. Daarom worden (de XML-tags van) deze namen uniek gemaakt door de naam van het gerelateerde entiteittype eraan vast te plakken. Hieronder een paar voorbeelden van dit mechanisme:

<element name="heeftVerplichte**Informatieobjecttype**" type="ztc:STTIOT-basis"   
 minOccurs="0" maxOccurs="unbounded"/>

<element name="heeftVerplichte**Eigenschap**" type="ztc:STTEIG-basis"   
 minOccurs="0" maxOccurs="unbounded"/>

<element name="heeftVerplichte**ZaakObjecttype**" type="ztc:STTZOT-basis"   
 minOccurs="0" maxOccurs="unbounded"/>

Een andere mogelijkheid is om te eisen dat de XML-tags van de relatienamen altijd op voorhand uniek zijn in het informatiemodel.

## Basistype

<complexType name="[Relatie.mnemonic]-basis">

<annotation>

<documentation>[Relatie.naam]</documentation>  
 </annotation>

<sequence minOccurs="0">

<element name="gerelateerde"

type="[ns prefix][[5]](#footnote-6):[Relatie.target.mnemonic][***SuffixSB***(Relatie.target)]"   
 minOccurs="0"/>

[( Element | Groep )\*]?

[<element name="authentiek"   
 type="[stuf ns prefix]:StatusMetagegeven-basis"

minOccurs="0" maxOccurs="unbounded"/>]?

[<element name="inOnderzoek"

type="[stuf ns prefix]:StatusMetagegeven-basis"

minOccurs="0" maxOccurs="unbounded"/>]?

[<element ref="[stuf ns prefix]:tijdvakRelatie" minOccurs="0"/>]?

[<element ref="[stuf ns prefix]:tijdvakGeldigheid" minOccurs="0"/>]?

<element ref="[stuf ns prefix]:tijdstipRegistratie" minOccurs="0"/>

[<element ref="[stuf ns prefix]:extraElementen" minOccurs="0"/>

<element ref="[stuf ns prefix]:aanvullendeElementen" minOccurs="0"/>  
 ]?

[<element name="historieMaterieel"

type="[ns prefix]:[Relatie.mnemonic]-basis" minOccurs="0"

maxOccurs="unbounded"/>]?

[<element name="historieFormeel"

type="[ns prefix]:[Relatie.mnemonic]-basis" minOccurs="0"   
 maxOccurs="unbounded"/>]?

[<element name="historieFormeelRelatie"

type="[ns prefix]:[Relatie.mnemonic]-basis" minOccurs="0"/>]?

[Relatie\*]?

</sequence>

<attribute ref="[ns prefix]:entiteittype" fixed="[Relatie.mnemonic]"/>

<attributeGroup ref="[stuf ns prefix]:entiteit"/>

</complexType>

Voor de opname van de elementen authentiek en inOnderzoek gelden dezelfde regels als voor fundamentele entiteittypen.

Het element tijdvakRelatie wordt alléén opgenomen als er materiële historie is gedefinieerd op de relatie.

De elementen tijdvakGeldigheid, extraElementen en aanvullendeElementen worden alleen opgenomen als de relatie een association class heeft in het UGM, oftewel de relatie heeft eigenschappen (elementen, groepen of zelfs een geneste relatie).

De functie *SuffixSB(entiteittype)* geeft de suffix “-super” terug als het *entiteittype* een supertype is en anders geeft deze functie de suffix “-basis” terug.

In het bovenstaande schema wordt de binding [Relatie.target.mnemonic] gebruikt. Deze verwijst naar de mnemonic van het als target gerelateerde entiteittype van de relatie.

Een complex type voor een relatie kan ook vertalingen van (groeps)attribuut- en relatiesoorten bevatten als de relatiesoort een relatieklasse is. Sommige elementen zijn niet altijd aanwezig zoals wordt aangegeven door de […]? notatie. Deze elementen worden alleen opgenomen als er aan de volgende condities wordt voldaan:

|  |  |
| --- | --- |
| **Element** | **Voorwaarde voor opname in complex type voor relatiesoort** |
| historieMaterieel | Op minimaal één van de elementen of groepen van de relatie is materiële historie gedefinieerd. |
| historieFormeel | Op minimaal één van de elementen of groepen van de relatie is formele historie gedefinieerd. |
| historieFormeelRelatie | Op de relatie is formele historie gedefinieerd. |

## Matchgegevens

<complexType name="[Relatie.mnemonic]-matchgegevens">

<annotation>

<documentation>Matchgegevens van de relatie</documentation>

</annotation>

<complexContent>

<restriction base="[ns prefix]:[Relatie.mnemonic]-basis">

<sequence minOccurs="0">

<element name="gerelateerde"

type="[ns prefix]:[Relatie.target.mnemonic][***SuffixSK***(Relatie.target)]"/>

</sequence>

<attribute ref="[ns prefix]:entiteittype" fixed="[Relatie.mnemonic]"/>

<attribute name="scope" type="[stuf ns prefix]:StUFScope"   
 use="prohibited"/>

</restriction>

</complexContent>

</complexType>

De functie *SuffixSK(entiteittype)* geeft de suffix “-super” terug als de gerelateerde entiteit een supertype is. Anders geeft deze functie “-matchgegevens” terug.

## Voorbeeld

Hieronder een voorbeeld van de vertaling van de relatiesoort “maakt deel uit van” die gebruikt wordt om relaties te leggen tussen objecten van het type Besluit (BSL) en Catalogus (CAT).

<element name="maaktDeelUitVanCatalogus "

type="ztc:BSTCAT-basis"

minOccurs="0"

maxOccurs="1"/>

<complexType name="BSTCAT-basis">

<annotation>

<documentation>Relatie naar de catalogus waarvan het besluittype deel uit   
 maakt.  
 </documentation>  
 </annotation>

<sequence minOccurs="0">

<element name="gerelateerde" type="ztc:CAT-basis"   
 minOccurs="0"/>

<element ref="stuf:extraElementen" minOccurs="0"/>

<element ref="stuf:aanvullendeElementen" minOccurs="0"/>

</sequence>

<attribute ref="ztc:entiteittype" fixed="BSTCAT"/>

<attributeGroup ref="stuf:entiteit"/>

</complexType>

<complexType name="BSTCAT-matchgegevens">

<annotation>

<documentation>Matchgegevens van de relatie</documentation>

</annotation>

<complexContent>

<restriction base="ztc:BSTCAT-basis">

<sequence minOccurs="0">

<element name="gerelateerde" type="ztc:CAT-matchgegevens"/>

</sequence>

<attribute name="ztc:entiteittypeBSTCAT" use="required"/>

<attribute name="scope" type="StUF:StUFScope" use="prohibited"/>

</restriction>

</complexContent>

</complexType>

Omdat er voor deze relatie geen historie is gedefinieerd zijn de elementen “historieMaterieel”, “historieFormeel” en “historieFormeelRelatie” niet opgenomen in het complex type “BSTCAT-basis”.

# Supertype

Als een entiteittype een generalisatie (supertype) is dan wordt het anders vertaald dan een gewoon entiteittype. Er zijn twee situaties mogelijk:

* Het entiteittype is een supertype maar is zelf geen subtype
* Het entiteittype is zowel een supertype als een subtype (er is dan sprake van geneste overerving)

## Case: Entiteittype is supertype maar geen subtype

Er wordt als volgt een complexType met de sufffix “-super” gegenereerd:

<complexType name="[Entiteittype.mnemonic]-super">

<annotation>

<documentation>[Entiteittype.naam]</documentation>

</annotation>

<sequence>

[(Element | Groep | Relatie)\*]

</sequence>

<attributeGroup ref="[stuf ns prefix]:entiteit"/>

</complexType>

Dit is hetzelfde als een -basis type maar dan zonder het attribute “entiteittype”.

## Case: Entiteittype is zowel supertype als subtype

Er wordt als volgt een complexType met de sufffix “-super” gegenereerd:

<complexType name="[Entiteittype.mnemonic]-super">

<annotation>

<documentation>[Entiteittype.naam]</documentation>

</annotation>

<complexContent>

<extension base="BG:[*Supertype*(Entiteittype.mnemonic)]-super">

<sequence>

[(Element | Groep | Relatie)\*]

</sequence>

</extension>

</complexContent>

</complexType>

## Vervolgstap

Als het supertype concreet is (niet abstract) dan wordt er ook een -basis type gegenereerd waarin het attribute “entiteittype” ingevuld met de mnemonic van het entiteittype door middel van extension is toegevoegd:

<complexType name="[Entiteittype.mnemonic]-basis">

<annotation>

<documentation>[Entiteittype.naam]</documentation>

</annotation>

<complexContent>

<extension base="[ns prefix]:[Entiteittype.mnemonic]-super">

<attribute ref="[ns prefix]:entiteittype" fixed="[Entiteittype.mnemonic]"/>

</extension>

</complexContent>

</complexType>

Van een supertype hoeven geen –matchgegevens type gegenereerd te worden. Dat gebeurt pas op het niveau van de subtypes, zie volgende sectie.

# Subtype

Als een entiteittype een subtype is maar zelf geen supertype (oftewel het blad of eindpunt van de inheritance-boom), dan wordt het als volgt vertaald als een extension op het supertype:

<complexType name="[Entiteittype.mnemonic]-basis">

<annotation>

<documentation>[Entiteittype.naam]</documentation>

</annotation>

<complexContent>

<extension base="BG:[*Supertype*(Entiteittype.mnemonic)]-super">

<sequence>

[(Element | Groep)\*]

<element name="gebeurtenis"   
 type="[stuf ns prefix]:GebeurtenisMetAttributes-basis"

minOccurs="0" maxOccurs="unbounded"/>

[<element name="authentiek"   
 type="[stuf ns prefix]:StatusMetagegeven-basis"

minOccurs="0" maxOccurs="unbounded"/>]?

[<element name="inOnderzoek"

type="[stuf ns prefix]:StatusMetagegeven-basis"

minOccurs="0" maxOccurs="unbounded"/>]?

<element ref="[stuf ns prefix]:tijdvakGeldigheid" minOccurs="0"/>

<element ref="[stuf ns prefix]:tijdstipRegistratie" minOccurs="0"/>

<element ref="[stuf ns prefix]:extraElementen" minOccurs="0"/>

<element ref="[stuf ns prefix]:aanvullendeElementen" minOccurs="0"/>

[<element name ="historieMaterieel"

type="[ns prefix]:[Entiteittype.mnemonic]-basis"

minOccurs="0"

maxOccurs="unbounded"/>]?

[<element name="historieFormeel"

type="[ns prefix]:[Entiteittype.mnemonic]-basis"

minOccurs="0" maxOccurs=”unbounded”/>]?

[Relatie\*]

</sequence>

<attribute ref="[ns prefix]:entiteittype" fixed="[Entiteittype.mnemonic]"/>

</extension>

</complexContent>

</complexType>

Alleen als één van de elementen of groepen een ‘Indicatie authentiek’ gelijk aan Authentiek, Landelijk kerngegeven, Gemeentelijk kerngegeven of Overig heeft in het subtype of de supertypen waarvan het subtype overerft, dan wordt het element “authentiek” opgenomen in bovenstaand complexType.

Alleen als één van de elementen of groepen een ‘Indicatie in onderzoek’ gelijk aan ‘Ja’ heeft in het subtype of de supertypen waarvan het subtype overerft, dan wordt het element “inOnderzoek” opgenomen in bovenstaand complexType. Als de waarde van ‘Indicatie in onderzoek’ gelijk is aan ‘Zie groep’ dan wordt het element “inOnderzoek” alleen opgenomen als de waarde in de groep gelijk aan ‘Ja’ is.  
  
Alleen als er op minimaal één van de elementen of groepen in het subtype of de supertypen waarvan het subtype overerft materiële historie is gedefinieerd door middel van het attribute meta:materieleHistorie=”true”, dan moet in het bovenstaande schema het “historieMaterieel” aanwezig zijn. Hetzelfde verhaal geldt voor het element “historieFormeel”.

De matchgegevens van het subtype worden als volgt gegenereerd:

<complexType name="[Entiteittype.mnemonic]-matchgegevens">

<complexContent>

<restriction base="[ns prefix]:[Entiteittype.mnemonic]-basis">

<sequence>

[(Element | Groep | Relatie)\*]

</sequence>

<attribute ref="[ns prefix]:entiteittype" fixed="[Entiteittype.mnemonic]  
 use="required"/>

<attribute name="scope" type="[stuf ns prefix]:StUFScope"   
 use="prohibited"/>

</restriction>

</complexContent>

</complexType>

Alleen die elementen, groepen en relaties in de restriction van het bovenstaande complexType die behoren tot de matchgegevens ( ‘Indicatie kerngegeven’ is gelijk aan ‘Ja’) worden gegenereerd. Bovendien kunnen de elementen, groepen en relaties afkomstig zijn uit een hiërarchie van één of meer supertypen. Deze moeten in de volgorde van overerving staan anders zal het leiden tot een validatiefout van XML Schema.

# Appendix A: Regels voor materiële en formele historie

## Regels voor fundamentele entiteiten

|  |  |
| --- | --- |
| F1 | Binnen elke fundamentele entiteit zijn de volgende elementen aanwezig:   * tijdstipRegistratie * tijdvakGeldigheid |
| F2 | Als voor een fundamentele entiteit geldt:   * er is materiële historie gedefinieerd op minimaal één van de eigenschappen   Dan zijn de volgende elementen aanwezig:   * historieMaterieel |
| F3 | Als voor een fundamentele entiteit geldt:   * er is formele historie gedefinieerd op minimaal één van de eigenschappen   Dan zijn de volgende elementen aanwezig:   * historieFormeel |

In regel F1 gaan we ervan uit dat de elementen tijdstipRegistratie en tijdvakGeldigheid ook gebruikt kunnen worden in een context zonder historie.

## Regels voor relatie-entiteiten

|  |  |
| --- | --- |
| R1 | Binnen elke relatie-entiteit zijn de volgende elementen aanwezig:   * tijdstipRegistratie |
| R2 | Als voor een relatie-entiteit geldt:   * heeft één of meer eigenschappen   Dan zijn de volgende elementen aanwezig:   * tijdvakGeldigheid |
| R3 | Als voor een relatie-entiteit geldt:   * er is materiële historie gedefinieerd op de relatie   Dan zijn de volgende elementen aanwezig:   * tijdvakRelatie |
| R4 | Als voor een relatie-entiteit geldt:   * er is formele historie gedefinieerd op de relatie   Dan zijn de volgende elementen aanwezig:   * historieFormeelRelatie |
| R5 | Als voor een relatie-entiteit geldt:   * er is materiële historie gedefinieerd op minimaal één van de eigenschappen   Dan zijn de volgende elementen aanwezig:   * historieMaterieel |
| R6 | Als voor een relatie-entiteit geldt:   * er is formele historie gedefinieerd op minimaal één van de eigenschappen   Dan zijn de volgende elementen aanwezig:   * historieFormeel |

In regels R1 en R2 gaan we ervan uit dat de elementen tijdstipRegistratie en tijdvakGeldigheid ook gebruikt kunnen worden in een context zonder historie.

1. Let op: de lege string moet meegegenereerd worden! [↑](#footnote-ref-1)
2. Zie ook GAB: <http://www.noraonline.nl/images/noraonline/f/fd/151902-harmonistatievoorstel-postcodes-v1.01.pdf> [↑](#footnote-ref-3)
3. We gaan er van uit dat alle complex-datatypes van zich zelf leeg kunnen zijn en dat nillable=”true” niet nodig is. [↑](#footnote-ref-4)
4. De functie minVoorkomens is hier voor de volledigheid opgenomen maar wordt nergens gebruikt. [↑](#footnote-ref-5)
5. Let wel: Het moet hier de namespace zijn van de gerelateerde en niet van de relatie. Binnen zkn0320 kan de gerelateerde zitten in bg0320 of in ztc0320. [↑](#footnote-ref-6)